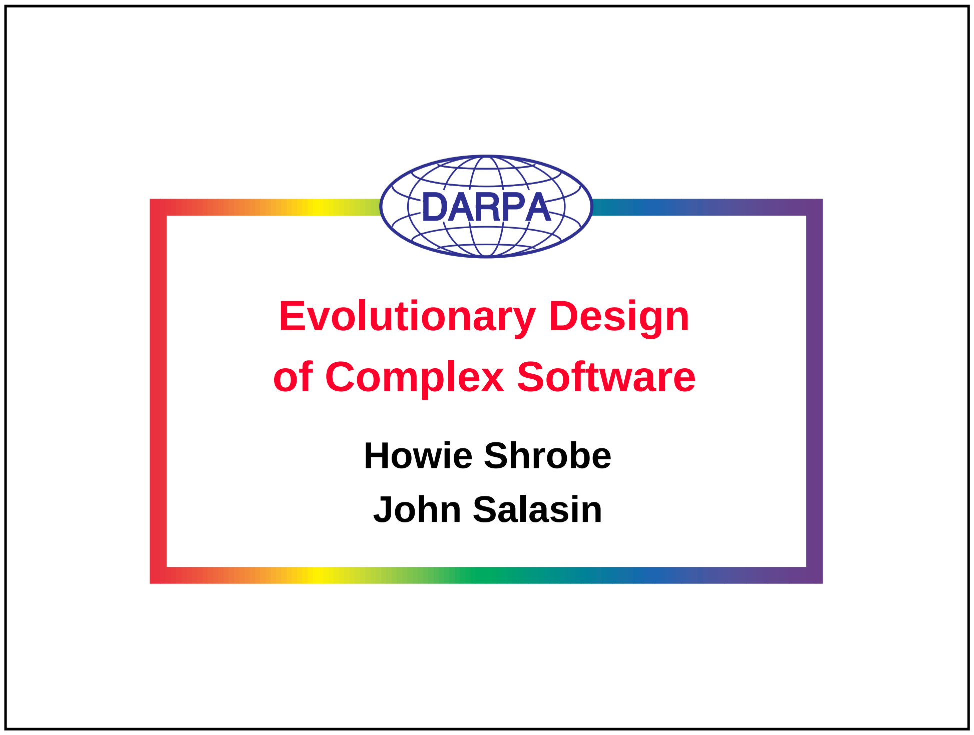
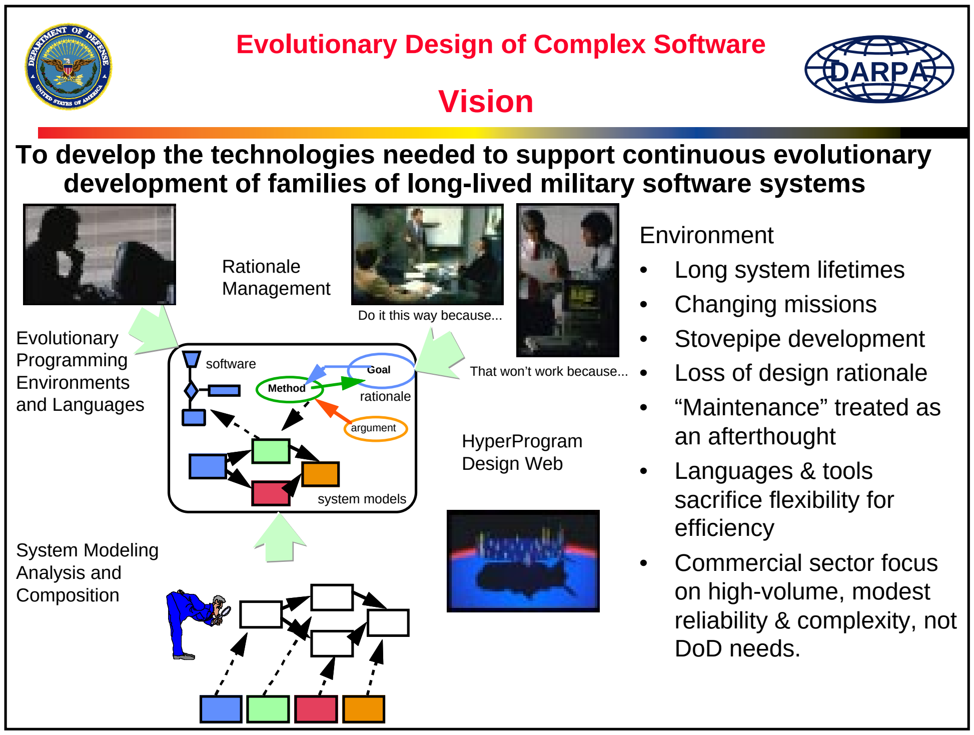


DARPA

Evolutionary Design of Complex Software

Howie Shrobe
John Salasin



Evolutionary Design of Complex Software

Vision

To develop the technologies needed to support continuous evolutionary development of families of long-lived military software systems

Environment

- Long system lifetimes
- Changing missions
- Stovepipe development
- Loss of design rationale
- "Maintenance" treated as an afterthought
- Languages & tools sacrifice flexibility for efficiency
- Commercial sector focus on high-volume, modest reliability & complexity, not DoD needs.

Rationale Management
Do it this way because...
That won't work because...
Evolutionary Programming Environments and Languages
software
Method
Goal
rationale
argument
system models
HyperProgram Design Web
System Modeling Analysis and Composition

Software is not only a very critical component of most DoD systems today, but it is also that part of the system which should be able to quickly adapt to changes in the system's environment.

Over the long period of time between the first formulation of system requirements to the eventual retirement of the system, many environmental changes are experienced -- changes to hardware and configuration; changes to mission; changes to operational elements; changes in amounts and types of data processed. Today we adapt to these changes by changing the software, but the changes are made only at considerable risk.

Change activity can all too often be characterized by:

•difficulty in making changes to an operational system without incurring costs disproportionate to the size of the change.

•an imperfect understanding of the current system.

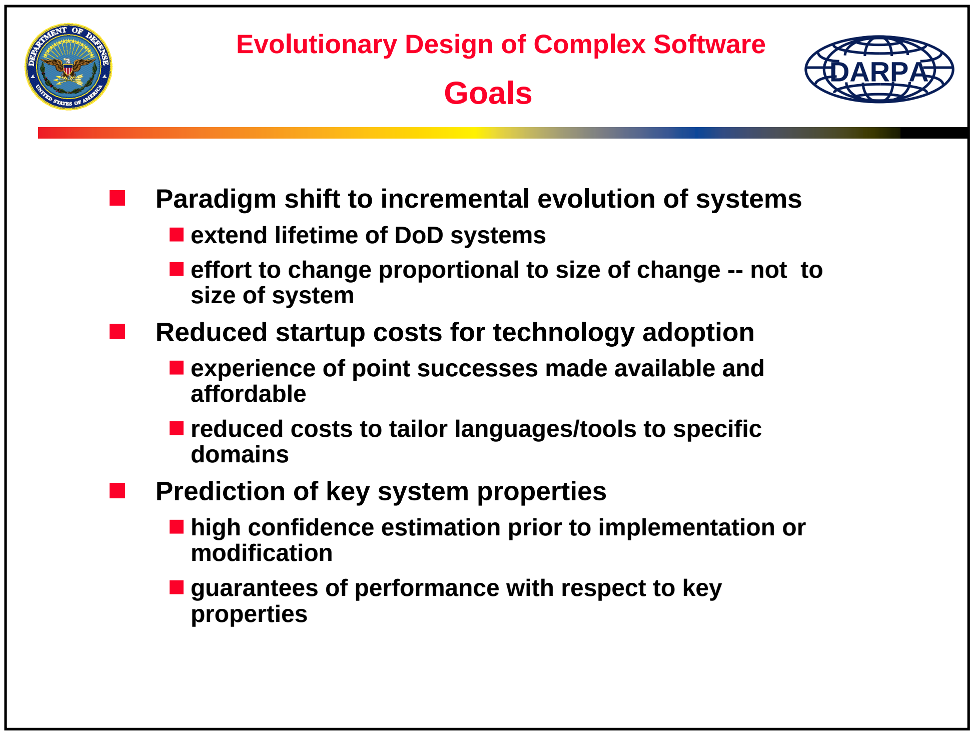
•an inability to accurately estimate the effort and resources required to make the change.

•small (conceptual) changes that affect large amounts of software.

•inability to predict with confidence the impact of a change before making it

•uncertainty of the validity of the change once it is made.

This program envisions a paradigm shift away from the specify- build-then-maintain life cycle assumed in the past to one of continuous evolution.



Evolutionary Design of Complex Software

Goals

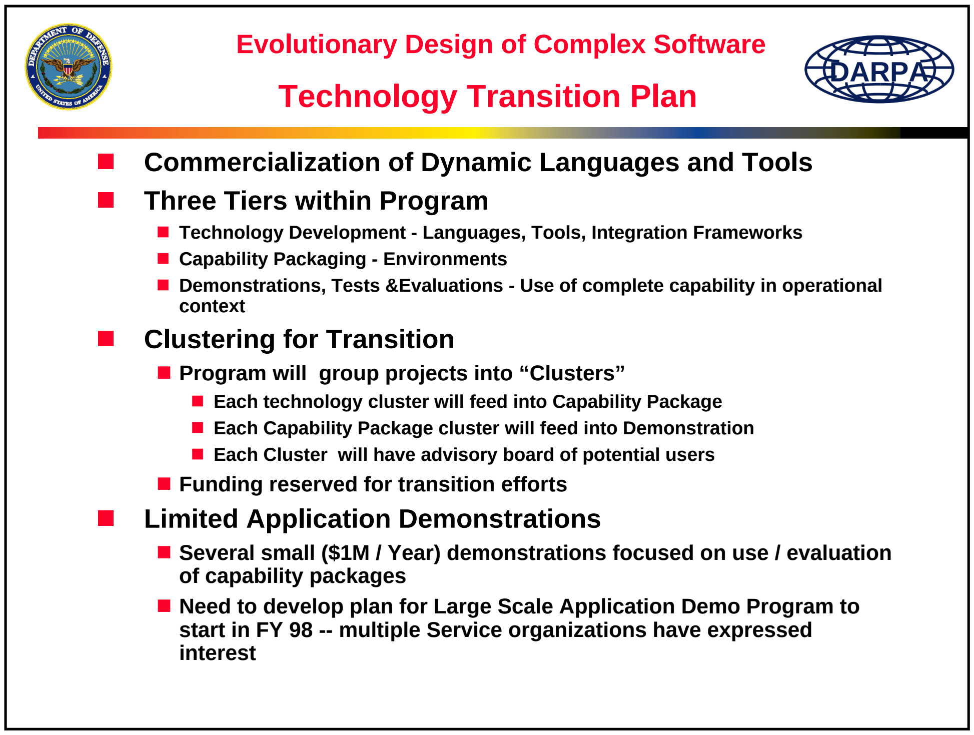
- Paradigm shift to incremental evolution of systems
  - extend lifetime of DoD systems
  - effort to change proportional to size of change -- not to size of system
- Reduced startup costs for technology adoption
  - experience of point successes made available and affordable
  - reduced costs to tailor languages/tools to specific domains
- Prediction of key system properties
  - high confidence estimation prior to implementation or modification
  - guarantees of performance with respect to key properties

The goals of EDCS are to provide **economic** methods for systems to keep up with changing requirements over their lifetimes by:

1. Providing a strong information base for evolution -- e.g., by supporting the capture, modular structuring, and effective access of design rationale information (in both formal and informal formats); managing relationships among all different aspects of information; and providing enhanced automated support for software/system understanding.

2. Enabling analysis of impacts of intended changes -- e.g., through static analysis of impacts of change on performance, function, and other system attributes such as reliability and safety; dynamic analysis of change through modeling, rapid prototyping and improved testing capabilities.

3. Enabling design and implementation of more adaptable systems - - e.g., through the use of: improved system and software architecture notations and representations; technology to effect system changes through changes to a (hardware or software) architectural description; incremental verification and validation; very high level domain specific languages; architecture-based component selection and code generation; reengineering of legacy systems; and use of dynamic implementation languages that enhance the capability to make changes to operational systems.



Evolutionary Design of Complex Software

Technology Transition Plan

- Commercialization of Dynamic Languages and Tools
- Three Tiers within Program
  - Technology Development - Languages, Tools, Integration Frameworks
  - Capability Packaging - Environments
  - Demonstrations, Tests &Evaluations - Use of complete capability in operational context
- Clustering for Transition
  - Program will group projects into “Clusters”
    - Each technology cluster will feed into Capability Package
    - Each Capability Package cluster will feed into Demonstration
    - Each Cluster will have advisory board of potential users
  - Funding reserved for transition efforts
- Limited Application Demonstrations
  - Several small ($1M / Year) demonstrations focused on use / evaluation of capability packages
  - Need to develop plan for Large Scale Application Demo Program to start in FY 98 -- multiple Service organizations have expressed interest

Several approaches are being taken to assure the transition of research results.

To increase the integration and usability of results, participants are divided into five technical thrusts, which have been termed "cluster” areas. The clusters are:

1) Rationale Capture and Software Understanding;

2) Architecture and Generation;

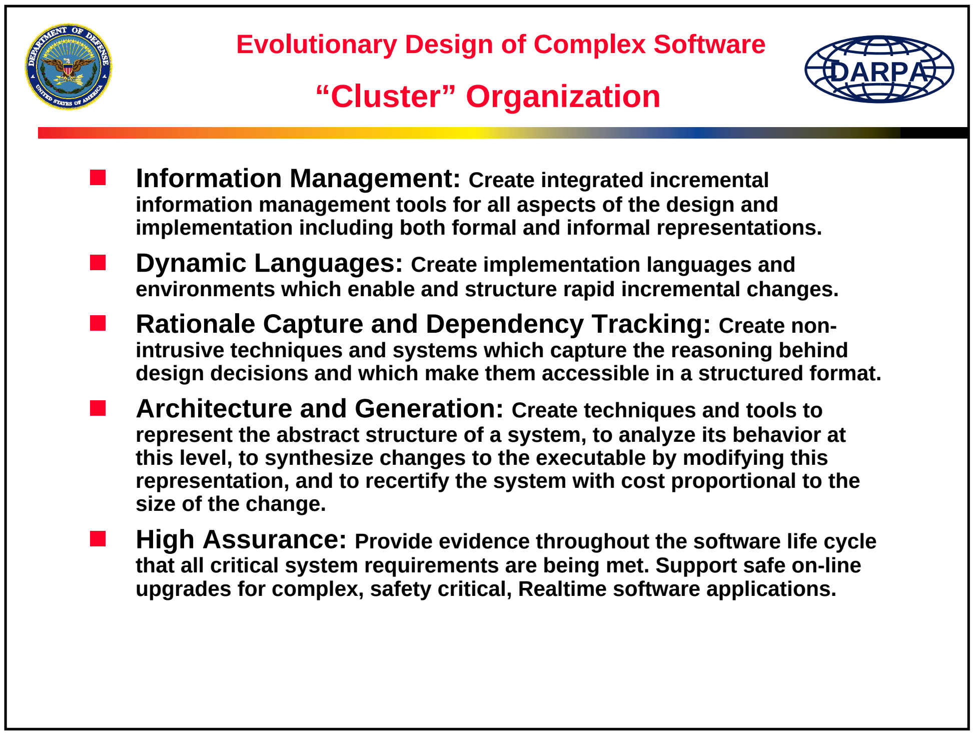
3) High Assurance and Real-Time;

4) Information Management , and;

5) Dynamic Languages.

Each cluster, as well as individual projects, are expected to adopt a "feature driven" approach. They are providing user oriented descriptions or fact sheets that identify the features or capabilities being developed. Clusters must define "usage scenarios" and coordinate intracluster demonstrations. Community use of EDCS results (i.e., "eating your own dog food") and close coordination are key parts of the program’s success formula.

The program is, in addition, conducting several small cross-cluster demonstrations on real military systems (e.g., F-16, B-2 upgrade, JSIMS, Satellite Ground Station).


Evolutionary Design of Complex Software

"Cluster" Organization

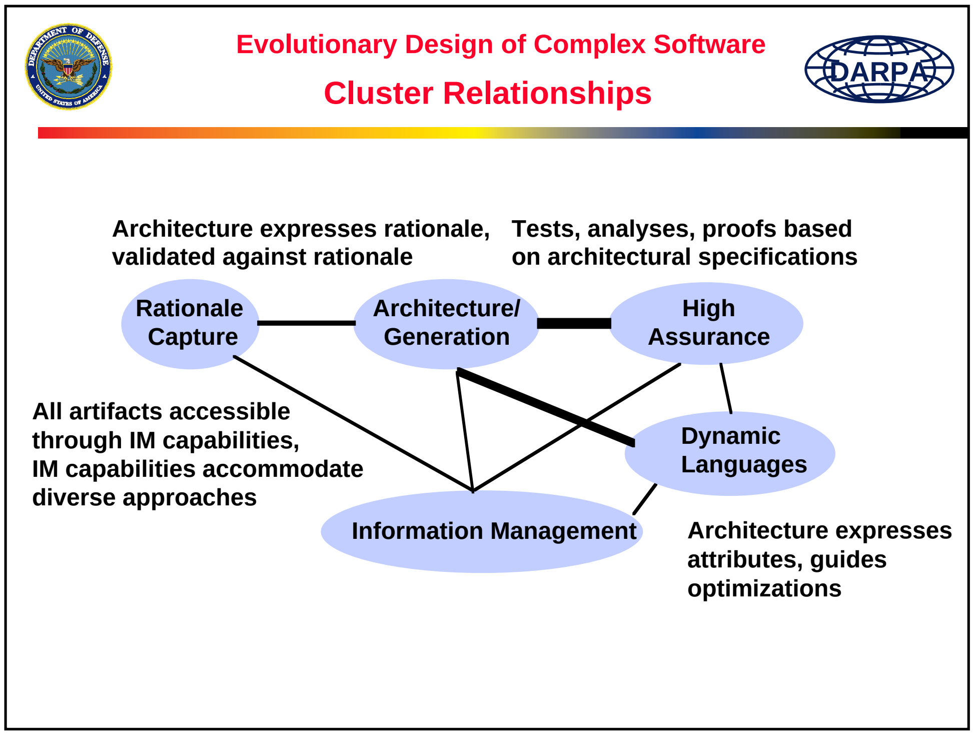
- **Information Management:** Create integrated incremental information management tools for all aspects of the design and implementation including both formal and informal representations.
- **Dynamic Languages:** Create implementation languages and environments which enable and structure rapid incremental changes.
- **Rationale Capture and Dependency Tracking:** Create non-intrusive techniques and systems which capture the reasoning behind design decisions and which make them accessible in a structured format.
- **Architecture and Generation:** Create techniques and tools to represent the abstract structure of a system, to analyze its behavior at this level, to synthesize changes to the executable by modifying this representation, and to recertify the system with cost proportional to the size of the change.
- **High Assurance:** Provide evidence throughout the software life cycle that all critical system requirements are being met. Support safe on-line upgrades for complex, safety critical, Realtime software applications.

Each cluster contains a mix of projects. "Product Development and Integration Projects" will be producing and integrating languages and tools. "Concept Development Projects" are smaller and more theoretical efforts, many co-funded by NSF. This categorization, as with the assignment of projects to clusters, is an informal arrangement designed to facilitate the transition of technology from basic research to end users.

In addition to supporting technology investigations and demonstrations of effectiveness in real military systems, the EDCS concept includes a focus on capability packaging. The primary purpose of the capability packaging activity is to provide demonstrations with scaled-up, evaluated, and integrated capabilities. It is intended that, within a specific area of concern, packages may be configured and tuned in many different ways including the selection from alternative technologies. Therefore, the packages, rather than constituting solutions, should be thought of as enabling easy creation of solutions. A Capability Package should not focus exclusively on a single technology / tool (e.g., a single ADL, a single reverse engineering tool) but should provide the ability to integrate multiple technologies / tools to solve a focused set of problems. It must be "open".

Part of the EDCS program is performing fundamental research aimed at developing formal verification technology. These programs are closely related to Assurance and Integration, a subprogram of the Information Survivability research area.

The following slides provide more detail on the clusters.

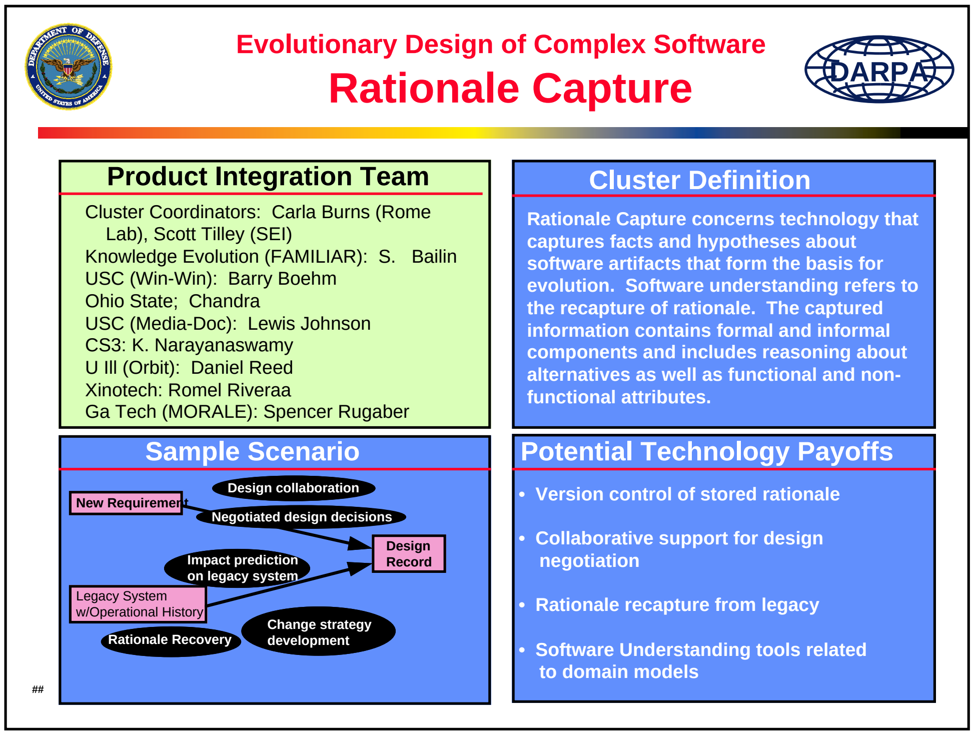


Evolutionary Design of Complex Software

Cluster Relationships

Architecture expresses rationale, validated against rationale
Tests, analyses, proofs based on architectural specifications
Rationale Capture
Architecture/ Generation
High Assurance
All artifacts accessible through IM capabilities, IM capabilities accommodate diverse approaches
Dynamic Languages
Information Management
Architecture expresses attributes, guides optimizations

Note that projects can be active in more than one cluster and that the concerns of the clusters overlap. This diagram shows some of the major interrelationships.



Evolutionary Design of Complex Software

Rationale Capture

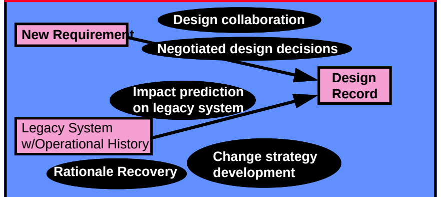
Product Integration Team

Cluster Coordinators: Carla Burns (Rome Lab), Scott Tilley (SEI)
Knowledge Evolution (FAMILIAR): S. Bailin
USC (Win-Win): Barry Boehm
Ohio State; Chandra
USC (Media-Doc): Lewis Johnson
CS3: K. Narayanaswamy
U Ill (Orbit): Daniel Reed
Xinotech: Romel Riveraa
Ga Tech (MORALE): Spencer Rugaber

Cluster Definition

Rationale Capture concerns technology that captures facts and hypotheses about software artifacts that form the basis for evolution. Software understanding refers to the recapture of rationale. The captured information contains formal and informal components and includes reasoning about alternatives as well as functional and non-functional attributes.

Sample Scenario


Design collaboration
New Requirement
Negotiated design decisions
Design Record
Impact prediction on legacy system
Legacy System w/Operational History
Change strategy development
Rationale Recovery

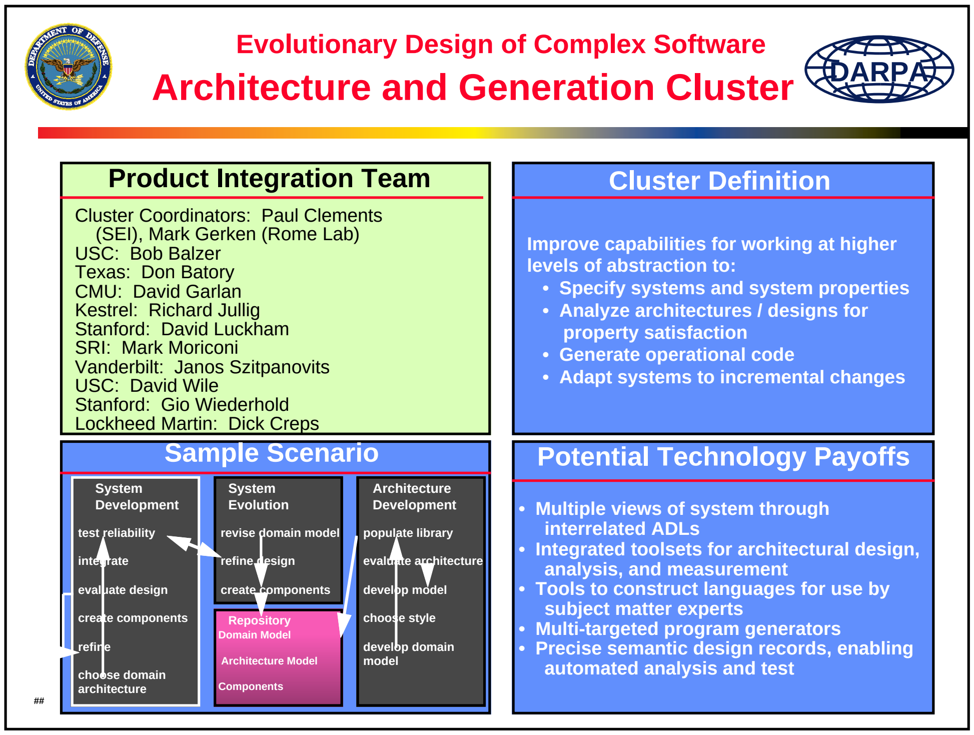
##

Potential Technology Payoffs

- Version control of stored rationale
- Collaborative support for design negotiation
- Rationale recapture from legacy
- Software Understanding tools related to domain models

This cluster addresses the software maintenance and reengineering problem by providing capabilities to initially capture or recover/generate explanations for design decisions and to use this knowledge to assists in making decisions affecting system evolution. This information is envisioned to be part of a software component's "design record." Research topics include: domain analysis, reverse engineering, consistency management, conflict resolution, and requirements engineering.

The usage scenario includes integrating new requirements with designs and rationale that are recovered from legacy systems. It provides capabilities to support collaborative work and to structure studies that lead to design decisions -- allowing both the decisions and the rationale behind the decisions to be captured. It integrates information about risks, technology, and architecture to improve our ability to predict cost and performance. Finally, it provides automated tools to tailor the explanation of software to the particular background of a user and the task he or she is performing.



Evolutionary Design of Complex Software

Architecture and Generation Cluster

Product Integration Team

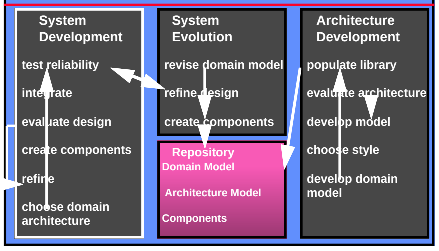
Cluster Coordinators: Paul Clements (SEI), Mark Gerken (Rome Lab)
USC: Bob Balzer
Texas: Don Batory
CMU: David Garlan
Kestrel: Richard Jullig
Stanford: David Luckham
SRI: Mark Moriconi
Vanderbilt: Janos Szitpanovits
USC: David Wile
Stanford: Gio Wiederhold
Lockheed Martin: Dick Creps

Cluster Definition

Improve capabilities for working at higher levels of abstraction to:

- **Specify systems and system properties**
- **Analyze architectures / designs for property satisfaction**
- **Generate operational code**
- **Adapt systems to incremental changes**

Sample Scenario



Potential Technology Payoffs

- **Multiple views of system through interrelated ADLs**
- **Integrated toolsets for architectural design, analysis, and measurement**
- **Tools to construct languages for use by subject matter experts**
- **Multi-targeted program generators**
- **Precise semantic design records, enabling automated analysis and test**

##

This cluster focus is on the various roles that a software architecture plays in the initial specification and design of a system, as well as how it can support its evolution through its adaptability, extendibility, and scalability. Research topics include:

•architecture notation and representation,

•domain-specific modeling,

•architecture description languages (and their interoperability),

•styles and patterns,

•static analysis (e.g., constraint satisfaction, views and visualization), dynamic analysis,

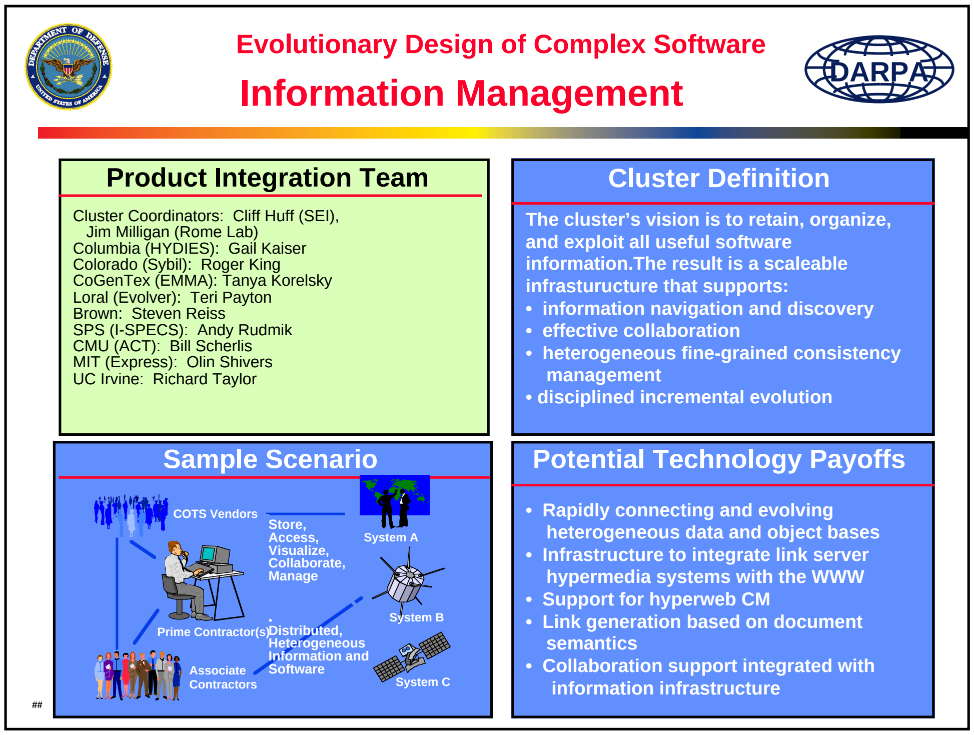
•configuration support,

•composition assistance,

•generation techniques, and

•architecture-centered processes.

More basic research efforts are focusing on specifying the semantics of architecture description languages and on developing improved ways of representing constraints and using the constraints to supports system evolution.



Evolutionary Design of Complex Software

Information Management

Product Integration Team

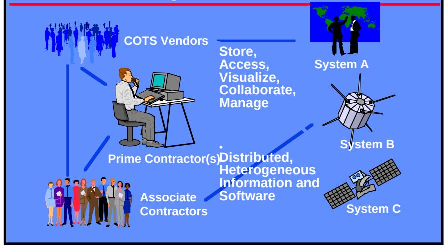
Cluster Coordinators: Cliff Huff (SEI), Jim Milligan (Rome Lab)
Columbia (HYDIES): Gail Kaiser
Colorado (Sybil): Roger King
CoGenTex (EMMA): Tanya Korelsky
Loral (Evolver): Teri Payton
Brown: Steven Reiss
SPS (I-SPECS): Andy Rudmik
CMU (ACT): Bill Scherlis
MIT (Express): Olin Shivers
UC Irvine: Richard Taylor

Cluster Definition

The cluster's vision is to retain, organize, and exploit all useful software information.The result is a scaleable infrasturucture that supports:

- information navigation and discovery
- effective collaboration
- heterogeneous fine-grained consistency management
- disciplined incremental evolution

Sample Scenario


COTS Vendors
Store, Access, Visualize, Collaborate, Manage
System A
System B
Prime Contractor(s)
Distributed, Heterogeneous Information and Software
Associate Contractors
System C

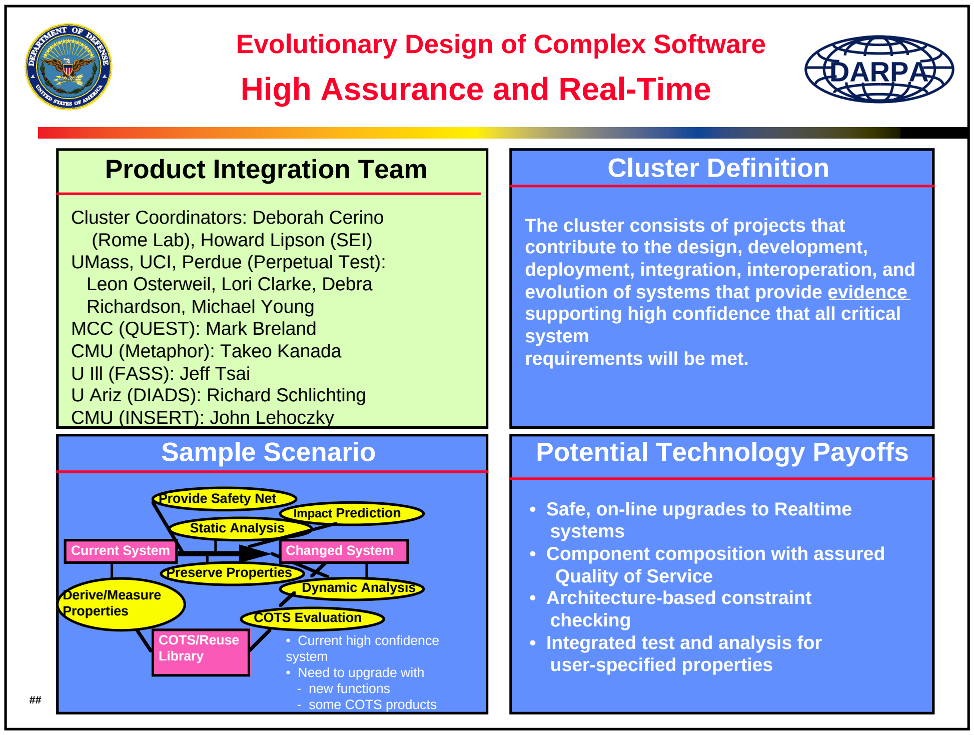
##

Potential Technology Payoffs

- Rapidly connecting and evolving heterogeneous data and object bases
- Infrastructure to integrate link server hypermedia systems with the WWW
- Support for hyperweb CM
- Link generation based on document semantics
- Collaboration support integrated with information infrastructure

One of the problems faced in evolving systems is the lack of useful information about the system and the history of its development / evolution. Even the information that generally is available, such as source code, documentation, and test data, is difficult to assimilate and may be inconsistent and inaccurate. Other information crucial to effecting system changes such as the rationale behind some of the original design choices is almost never available. In an evolutionary system, this information will be captured and made accessible throughout the system's lifetime.

This cluster's goal is to provide an advanced information substrate to support the conceptualization, representation, and manipulation of multi-media software artifacts. Research topics include: the WWW, CORBA, integration mechanisms, versioning, hyperlinks, semantic links, evolution of persistent data, information models, views, metalevel information representation and semantics, and process and transaction enactment.



Evolutionary Design of Complex Software

High Assurance and Real-Time

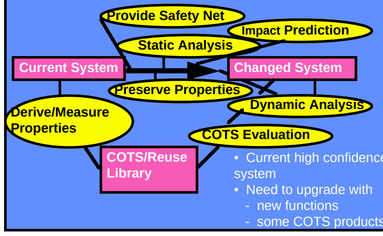
Product Integration Team

Cluster Coordinators: Deborah Cerino (Rome Lab), Howard Lipson (SEI)
UMass, UCI, Perdue (Perpetual Test): Leon Osterweil, Lori Clarke, Debra Richardson, Michael Young
MCC (QUEST): Mark Breland
CMU (Metaphor): Takeo Kanada
U Ill (FASS): Jeff Tsai
U Ariz (DIADS): Richard Schlichting
CMU (INSERT): John Lehoczky

Cluster Definition

The cluster consists of projects that contribute to the design, development, deployment, integration, interoperation, and evolution of systems that provide evidence supporting high confidence that all critical system
requirements will be met.

Sample Scenario



- Current high confidence system
- Need to upgrade with
  - new functions
  - some COTS products

Potential Technology Payoffs

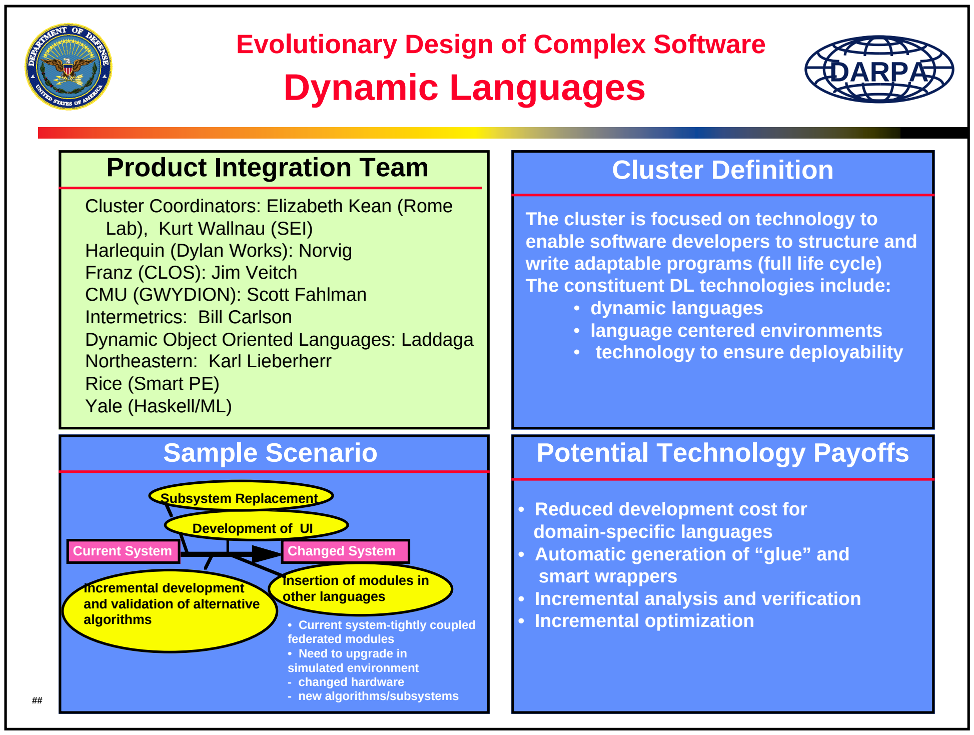
- Safe, on-line upgrades to Realtime systems
- Component composition with assured Quality of Service
- Architecture-based constraint checking
- Integrated test and analysis for user-specified properties

##

This cluster addresses the topic of providing evidence throughout the software life cycle supporting high confidence that all critical system requirements will/are being met.

Research topics include: support for static analysis and testing with special emphasis on their synergistic support for software evolution; reducing time and effort spent on software testing while producing software of equal or better quality (via specification & modeling languages, quantitative quality metrics, multi-media aids, technique integration mechanisms); supporting safe on-line upgrades for complex, safety critical, realtime software applications; developing systematic and algorithmic methods for analyzing likely changes to software solutions and for localizing these changes; and representing behavioral and architectural specifications in a knowledge base and generating code from that representation.

Other basic research efforts are providing a formal basis for run-time assurance and program verification.



Evolutionary Design of Complex Software

Dynamic Languages

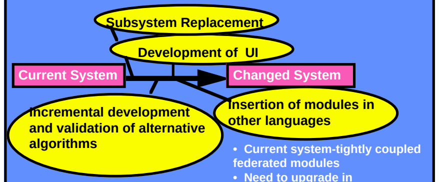
Product Integration Team

Cluster Coordinators: Elizabeth Kean (Rome Lab), Kurt Wallnau (SEI)
Harlequin (Dylan Works): Norvig
Franz (CLOS): Jim Veitch
CMU (GWYDION): Scott Fahlman
Intermetrics: Bill Carlson
Dynamic Object Oriented Languages: Laddaga
Northeastern: Karl Lieberherr
Rice (Smart PE)
Yale (Haskell/ML)

Cluster Definition

The cluster is focused on technology to enable software developers to structure and write adaptable programs (full life cycle)
The constituent DL technologies include:
- **dynamic languages**
- **language centered environments**
- **technology to ensure deployability**

Sample Scenario


Subsystem Replacement
Development of UI
Current System
Changed System
Incremental development and validation of alternative algorithms
Insertion of modules in other languages

- **Current system-tightly coupled federated modules**
- **Need to upgrade in simulated environment**
- **changed hardware**
- **new algorithms/subsystems**

##

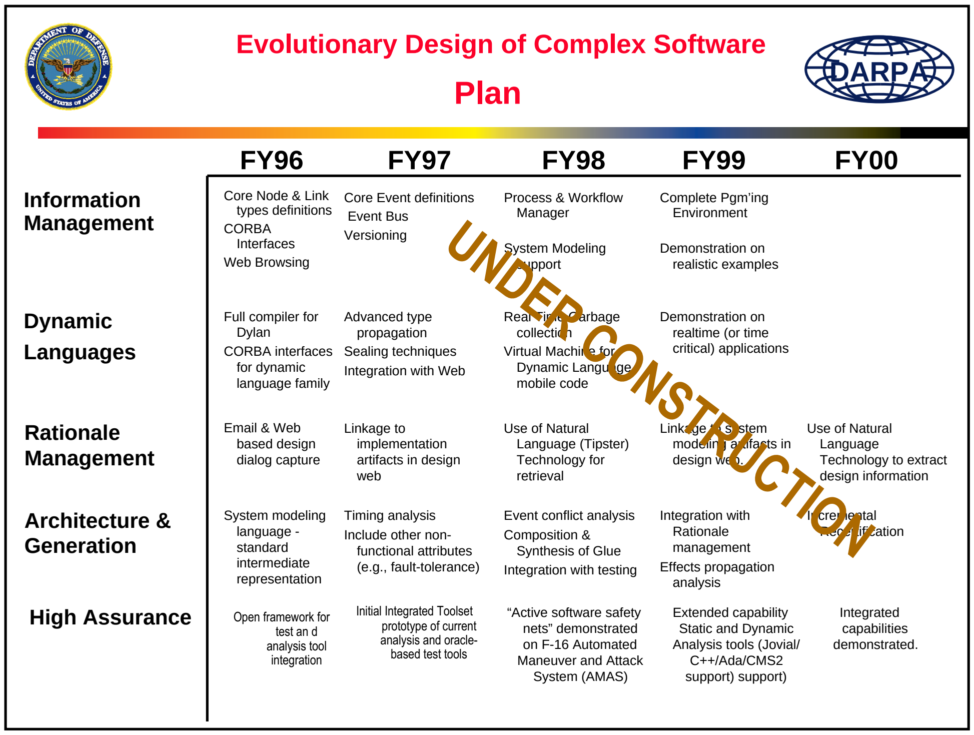
Potential Technology Payoffs

- **Reduced development cost for domain-specific languages**
- **Automatic generation of “glue” and smart wrappers**
- **Incremental analysis and verification**
- **Incremental optimization**

Evolution is characterized by adaptability. The software in most systems today is parameterized to some extent. That is, the designer provides for some limited variation in the software characteristics by preplanning some support for changing and/or tailoring it at runtime, at system generation time, or some other time in the system's lifecycle beyond its initial development. Thus, binding of the system (for these limited, pre-established parameters) is delayed and the system has been made adaptable. EDCS extends the adaptability of systems far beyond that usually experienced through software parameterization.

The use of dynamic languages both for their late binding characteristic and for their underlying support for the creation of very high level domain specific languages is an important area for EDCS research. Of particular interest are: the ability to achieve high performance applications using dynamic languages; the ability to apply advanced compilation techniques; the ability to selectively provide static as well as dynamic bindings; the ability to provide predictable and realtime performance; and the ability to intermix dynamic and conventional programming languages for a single application.

This cluster is providing an advanced software development environment for Dylan, Haskell, ML, Ada95, Java, and CLOS. This includes support of language interoperability. Research topics include: hyper-program structure, higher-level analysis tools, very high level objects, modular proofs of correctness, and aspectual decomposition.

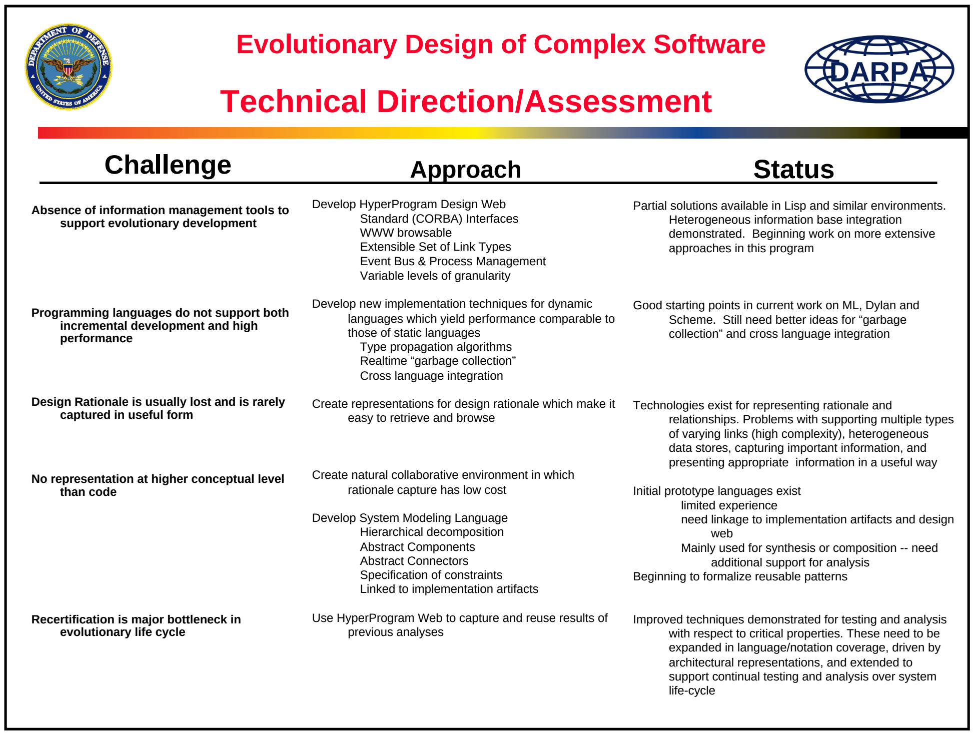


Evolutionary Design of Complex Software

Plan

	FY96	FY97	FY98	FY99	FY00
Information Management	Core Node & Link types definitions CORBA Interfaces Web Browsing	Core Event definitions Event Bus Versioning	Process & Workflow Manager System Modeling Support	Complete Pgm'ing Environment Demonstration on realistic examples	
Dynamic Languages	Full compiler for Dylan CORBA interfaces for dynamic language family	Advanced type propagation Sealing techniques Integration with Web	Real Time Garbage collection Virtual Machine for Dynamic Language mobile code	Demonstration on realtime (or time critical) applications	
Rationale Management	Email & Web based design dialog capture	Linkage to implementation artifacts in design web	Use of Natural Language (Tipster) Technology for retrieval	Linkage to system modeling artifacts in design web.	Use of Natural Language Technology to extract design information
Architecture & Generation	System modeling language - standard intermediate representation	Timing analysis Include other non-functional attributes (e.g., fault-tolerance)	Event conflict analysis Composition & Synthesis of Glue Integration with testing	Integration with Rationale management Effects propagation analysis	Incremental Recertification
High Assurance	Open framework for test an d analysis tool integration	Initial Integrated Toolset prototype of current analysis and oracle-based test tools	"Active software safety nets" demonstrated on F-16 Automated Maneuver and Attack System (AMAS)	Extended capability Static and Dynamic Analysis tools (Jovial/ C++/Ada/CMS2 support) support)	Integrated capabilities demonstrated.

The EDCS kickoff meeting was held 17-19 July 1996. As a followup to that meeting, participants are revising their project plans to support cluster activities and to define capability packages. This plan is being revised.



Evolutionary Design of Complex Software

Technical Direction/Assessment

Challenge	Approach	Status
Absence of information management tools to support evolutionary development	Develop HyperProgram Design Web Standard (CORBA) Interfaces WWW browsable Extensible Set of Link Types Event Bus & Process Management Variable levels of granularity	Partial solutions available in Lisp and similar environments. Heterogeneous information base integration demonstrated. Beginning work on more extensive approaches in this program
Programming languages do not support both incremental development and high performance	Develop new implementation techniques for dynamic languages which yield performance comparable to those of static languages Type propagation algorithms Realtime “garbage collection” Cross language integration	Good starting points in current work on ML, Dylan and Scheme. Still need better ideas for “garbage collection” and cross language integration
Design Rationale is usually lost and is rarely captured in useful form	Create representations for design rationale which make it easy to retrieve and browse Create natural collaborative environment in which rationale capture has low cost	Technologies exist for representing rationale and relationships. Problems with supporting multiple types of varying links (high complexity), heterogeneous data stores, capturing important information, and presenting appropriate information in a useful way
No representation at higher conceptual level than code	Develop System Modeling Language Hierarchical decomposition Abstract Components Abstract Connectors Specification of constraints Linked to implementation artifacts	Initial prototype languages exist limited experience need linkage to implementation artifacts and design web Mainly used for synthesis or composition -- need additional support for analysis Beginning to formalize reusable patterns
Recertification is major bottleneck in evolutionary life cycle	Use HyperProgram Web to capture and reuse results of previous analyses	Improved techniques demonstrated for testing and analysis with respect to critical properties. These need to be expanded in language/notation coverage, driven by architectural representations, and extended to support continual testing and analysis over system life-cycle